

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-312109

(43)Date of publication of application : 09.11.1999

(51)Int.Cl. G06F 12/00
G06F 12/00

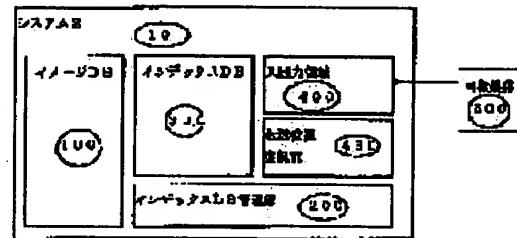
(21)Application number : 10-118125 (71)Applicant : HITACHI LTD
(22)Date of filing : 28.04.1998 (72)Inventor : MURAKAMI TATSUYA
UCHIYAMA TAKASHI

(54) RELATIONAL DATA BASE SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To readily move a part of a data base(DB) from a system for managing files by using a relational data base(RDB) to another system by providing the system with an I/O area for DB files and storing data of volume corresponding to a single carriable piece medium in the I/O area so as to always segment the data.

SOLUTION: When the part of a DB in a data management system using the RDB to another system is transferred, the partial DB separated and copied from the original DB cannot function as a DB. As a result when the data of a large scale system is transferred by using a carriable medium such as an MO, index data for the whole system are required to be transferred through a conventional method. Thereby the system is provided with the I/O area for inputting/outputting a data file to/from the RDB, and the DB file is transferred between systems by using the I/O area. Data files managed by the DB are recorded in the same carriable medium.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

[decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-312109

(43) 公開日 平成11年(1999)11月9日

(51) Int.Cl.⁶
G 0 6 F 12/00

識別記号
5 2 0
5 1 2

F I
G 0 6 F 12/00

5 2 0 A
5 1 2

審査請求 未請求 請求項の数 6 O.L (全 6 頁)

(21) 出願番号 特願平10-118125

(22) 出願日 平成10年(1998)4月28日

(71) 出願人 000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地
(72) 発明者 村上 達也
神奈川県小田原市国府津2880番地 株式会
社日立製作所ストレージシステム事業部内
(72) 発明者 内山 隆司
神奈川県小田原市国府津2880番地 株式会
社日立製作所ストレージシステム事業部内
(74) 代理人 弁理士 小川 勝男

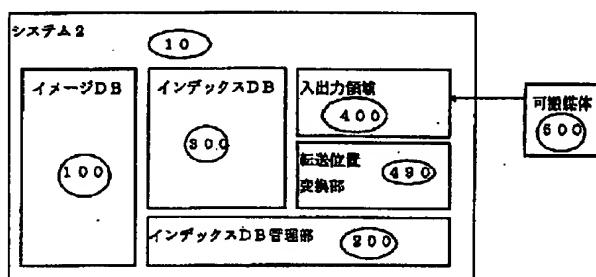
(54) 【発明の名称】 リレーションナルデータベースシステム

(57) 【要約】

【課題】 リレーションナルデータベース (RDB) を用いてファイル管理を行うシステムにおいて、DBのファイルとして入出力専用の領域を設け、可搬媒体1枚分に相当する量のデータを常に切り出せる形で保管することにより、DBの一部を容易に別のシステムに移動できる。

【解決手段】 RDBによるデータ管理システムでは、DBの一部を別のシステムに移動する場合に、単にDBの一部を切り離してコピーするだけではDBとして機能できない。このため大規模システムのデータをMO等の可搬媒体を用いて移動するには、インデックスデータはシステム全体分を移動しなければならなかった。RDBのデータファイルとして入出力専用の領域を設け、その領域を用いてシステム内外のDBファイルの受け渡しを行う。DBにより管理されたデータファイルは、同一の可搬媒体に記録する。

図1



【特許請求の範囲】

【請求項 1】データベースの一部を可搬媒体へ複製する機能を有し、該可搬媒体に複製されたデータベースの一部を、データベースとして利用できる機能を有するリレーションナルデータベースシステム。

【請求項 2】リレーションナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録するデータベースファイルと、該データの一部のみからなる入出力用のデータベースファイルと、可搬媒体へのデータ入出力手段と、該入出力用データベースの内容を該可搬媒体へ移動する手段を持つことを特徴とするリレーションナルデータベースシステム。

【請求項 3】リレーションナルデータベースシステムにおいて、管理しているデータベースの一部を登録する入出力用データベースを持つことを特徴とするリレーションナルデータベースシステム。

【請求項 4】請求項 2 のデータベースシステムにおいて、データを登録する場合にデータベースファイルと入出力用データベースファイルに同時に記録する手段を持つことを特徴とするリレーションナルデータベースシステム。

【請求項 5】リレーションナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録する複数のデータベースファイルと、登録されるデータに応じて登録先のデータベースを切り替える手段と、各データベースの内容をそれぞれ別の可搬媒体に移動する手段を有することを特徴とするデータベースシステム。

【請求項 6】リレーションナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録する複数のデータベースファイルと、複数の可搬媒体から該複数のデータベースファイルにデータを移動する手段と、該複数のデータベースを間を同一のユーザによりアクセスする手段を有することを特徴とするデータベースシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、電子計算機において多量のデータを管理するデータベースシステムに関し、特に、大量のデータファイルをデータベースを用いて管理するシステムにおいて、内部に蓄積された大規模なデータの一部を、可搬媒体上に複製して別のシステムに移動して利用することを可能とするデータ管理方法に関するものである。

【0002】

【従来の技術】計算機システムにおいて、大量のデータファイルを管理する場合、各データ毎にインデックスを付け、そのインデックスを用いて管理又は検索を行う方法が一般的である。

【0003】ここで、インデックスの管理には大別して

2種類の方法がある。一つは、各項目を予め定められた1つの表に蓄積し、各行単位で関連を持たせるテーブルにより管理する方法である。残る一つは、2項目のテーブルを複数組み合わせることで複雑な情報を記録するリレーションナルデータベースを用いる方法である。テーブルによりインデックスを管理した場合に、登録されているファイルの一部を切り出す際には、インデックスも該当する行の部分だけを切り出すことができる。一方、リレーションナルデータベースによりインデックスを管理した場合は、システムの構築後でもデータ構造を任意に変更することが容易である。

【0004】

【発明が解決しようとする課題】テーブル形式でインデックスを保存した場合には、可搬媒体上にファイルとそれに対応する部分のインデックスを書き込めば、容易に、单一媒体に対するデータの移動が行える。しかし、この方法では、複数媒体にまたがるような大規模なシステムで、各媒体のインデックスを一度に検索するような場合には、一旦、計算機システムに存在するワークファイル上で、全てのインデックス情報を合成する必要がある。このとき各媒体のテーブルは単に連続しているだけであり、規模の増加に伴い、検索には多大な時間を要することになる。

【0005】内容をソートした形で検索するためには、個々のインデックスを合成し、改めてソートしなおすような処理が必要である。また複数の媒体間でデータ構造が異なる場合は、それらを一括して管理することができない。

【0006】一方、リレーションナルデータベース（以後、RDBと略す。）では、複数のテーブルをたどりながら目的のデータを取得することになる。そのため、あるデータをたどるときには各テーブルで、そのデータに関連する情報の位置は異なったものとなる。したがって、データベース（以後、DBと略す。）上の一部のデータを切り出して、別のシステムに移動させようとした場合に、各テーブルから同一の位置に存在する情報の一部分だけを切り出しても、DBとしては機能できない。

【0007】イメージなどの容量の大きなファイルの管理にRDBを用いると、そのファイルの一部、例えば、イメージのデータ部分は、光磁気ディスク媒体（以後、MOと略す。）などにより容易に移動できるが、そのイメージに関連するインデックス情報は、RDB全体を1つの単位として移動しなければならない。このためイメージはMOに蓄積しながら、そのインデックスは、DATなどの大容量記憶媒体を用いて保存しなければならない事例が生じる。

【0008】また、RDBの容量が複数の媒体を必要とする場合には、そのうち一部の媒体だけではRDBの内容を復元することは不可能であり、必ず全ての媒体をまとめて移動する必要がある。この結果、既に多量のデータ

タファイルとそれを管理する大規模なインデックスDBを有するシステムに、外部から新たなデータを可搬媒体で追加する際に、インデックスデータを各データ毎に辿り、既存のDBへ個々に追加登録する、インポート処理が必要となる。

【0009】

【課題を解決するための手段】本発明では、上記の課題を解決するため、インデックスの管理にRDBを用いる。そしてインデックスの一部をデータファイルと共に移動するため、RDBに付属して小規模な入出力領域を予め用意する。また、この入出力領域に対してDBの管理プログラムには、大規模DBと同じ条件で接続する機能を持たせる。具体的には、同一のユーザ名を登録し、各ユーザに同一権限を与える等である。他のシステムより可搬媒体で移動してきたファイルと、RDB内部の情報（データ部分）は、そのまま上記の入出力領域にコピー（複製）する手段を持つ。

【0010】この結果、ここで入力されたファイルのインデックスは小規模な独立したRDBとして扱うことにより、既にユーザが有する大規模なDBへインポート処理を行わずとも媒体内部のデータを確認することが可能となる。

【0011】

【発明の実施の形態】本発明の利用分野として最も効果的なイメージ管理システムを例に、より詳しくは、RDBによるインデックス管理を用いた大規模なイメージ管理システムにおいて、そのデータの一部をMOなどの記憶媒体により移動することを例に、以下、図を用いて本発明の実施の形態の一例を説明する。イメージ管理システムでは、各イメージは個別のファイルとして扱われる。したがって、当該ファイルを可搬媒体にコピーすることにより、容易にシステム間の移動ができることが大きなメリットとなっている。そのため、既存のイメージ管理システムの多くは、イメージファイルと共に、そのファイルに対応したインデックスデータ同じ記憶媒体であるMOなどの光媒体に蓄積していた。別のシステム

（装置）にデータを移動する場合には、MO1枚を運ぶだけで、そのままデータを利用できるという機能を備えていた。

【0012】これに対して、近年提案されているシステムでは、インデックス情報をRDBで管理することにより汎用性及び検索機能の高いシステムを構築することを目的としている。

【0013】図1に本発明の全体図を示す。なお、本図では各機能をブロック化して表現しているが、実際には、ソフトウェアにて機能を実現することも可能である。100は、多数のイメージデータファイルを蓄積するイメージDB、200がデータベースの管理部分、300が該イメージデータのインデックス情報を管理するRDBのデータ部分、400がRDBの一部を可搬媒体

500に移動する際に用いる入出力領域のデータ部分、10がシステム全体を搭載したサーバを示す。

【0014】ここで、このシステムに蓄積されているイメージデータの総数を100万件とし、各イメージデータの量を100KB、1件当たりに付けられるインデックスデータを10KBとすると、このシステムに搭載されるイメージデータの総量は100GB、インデックスデータは10GBになり、データ1件当たりの蓄積に要するデータ量は110KBとなる。

【0015】本システムに接続する可搬媒体500の容量を2GBとすると、1枚の媒体に記録できるデータ数は約18000件になる。

【0016】入出力領域300はRDBである。その容量は、2GBの容量を有する可搬媒体500に搭載できる18000件分のインデックスを蓄積できる容量が必要であり、約200MBとなる。

【0017】はじめに別のシステム（システム1）で作成されたデータを可搬媒体500に登録し、本システム（システム2）に搬入した場合について説明する。図1で、10がシステム2にあたる。システム1で登録されたデータを可搬媒体500によりシステム2（10）に装填した場合、媒体中のインデックス領域を入出力領域400にコピーする。

【0018】図2にインデックスの内部のテーブルの構造の一例を示す。310が検索で用いるイメージファイルのタイトルと各ファイルに付けられたイメージIDを記録するタイトルテーブル、320が各IDとイメージファイルの名称を記録するファイル名称テーブルである。一方、DBはシステム自体の情報を管理するテーブルも持つ。350は本システムを利用する各ユーザを管理するためユーザ名とユーザのIDを記録するユーザテーブルである。

【0019】各イメージファイルは各ユーザ毎にアクセスの可否を設定する。330は各イメージ毎にアクセスを許すユーザのIDを記録したセキュリティテーブルである。

【0020】各システムは自身へのアクセス権限を持つユーザをユーザテーブルにあらかじめ登録しておく。本実施の態様では、2人のユーザ、“User1”と“User2”を例に説明する。ここで、テーブル350では“ユーザ1”にID“AAA”、“ユーザ2”にID“BBB”を関連付ける。

【0021】一方、イメージデータファイルの登録では、“文書1”的タイトルで登録する場合を例に各インデックスの内容を説明する。まず、タイトルテーブル310に、タイトル“文書1”とファイルのID“001”を記録する。この“文書1”を示すイメージデータのファイル名称“img1.xxx”をテーブル320に、イメージID“001”と共に記録する。各文書毎にどのユーザに対してアクセスを許可するかを記すのが、セキュ

リティーテーブル330である。この例では“文書1”つまりイメージID“001”が“User1”に対してのみアクセスを許可し、“文書2”つまりイメージID“002”に対しては、“User1”および“User2”共に許可することを示す。セキュリティーテーブル330には、各イメージIDとそのイメージへのアクセスを許可するユーザのIDが記録されている。ここでは説明を簡単にするために、セキュリティのレベルは“許可”か“否”かの1つだけであるが、実際には公知の方法により複数のレベルを設定できる。

【0022】データの移動を行う場合、システム2はシステム1が登録しているユーザを全て登録してある必要がある。つまり、システム1のユーザテーブル350とセキュリティーテーブル330の内容にはシステム2のユーザテーブルとセキュリティーテーブルの内容を全て含んでいることが必要である。

【0023】システム2は、通常は、RDBの管理部200がRDBデータ300に接続して動作し、イメージファイル100へのアクセスを制御する。システム1から可搬媒体500によりシステム2にデータを移動する場合には、データのうち上記各インデックスデータは出入力領域400にコピーされ、RDB管理部200はデータ300に代わり入出力領域400に接続して動作することで、一時的に可搬媒体500中のイメージデータにアクセスできるようになる。

【0024】このようにすればシステム2が、入出力領域400を介して、可搬媒体500に存在するインデックスデータにアクセスできるので、システム2のインデックスDB300に対して、別途、可搬媒体500からシステム2に対するインポート処理を施す必要がない。このため短時間でシステム1のDB内容（可搬媒体500）の確認が可能となる。

【0025】次に、図3を用いて転送位置変換部490を説明する。システム2が、RDBデータであると認識して、入出力領域400の内容をアクセスしている場合には、イメージデータの記録先は可搬媒体500の内部にある。システム1でイメージを登録した時点では、イメージデータはシステム1内に存在するため、そのままではイメージデータとの連携がとれない。

【0026】そのため、IDとイメージファイルの位置情報を記録したテーブル320において、変換を行う必要がある。しかしながら個々のデータを変換しながら入出力領域400に記録するのではなく、短時間の立ち上げが不可能となる。したがってサーバ10から入出力領域400へアクセスするときは、イメージファイルパスは先頭に可搬媒体の位置を自動で添付して応答するように、入出力領域400とDB管理部300の間に転送位置変換部490を有する。ここでは、イメージファイルのパス名を返す場合に、パスの先頭に可搬媒体500のパスを付け加えて送るものである。

【0027】転送位置変換部490（図3）の動作は次の通り。まず登録元のシステム1においてイメージデータ.xxxの位置が¥IMAGE¥FOLDER¥であったとする（320、図2）。このとき各イメージのパス名としては、¥IMAGE¥FOLDERに連続して、例えば¥IMAGE¥FOLDER¥img1.xxxxとしてテーブル320に記録される。これを可搬媒体500にコピーしシステム2に移動した場合には、イメージデータの位置はシステム2の¥IMAGE¥FOLDERになければならない。

【0028】しかしながらシステム2ではシステム2自身のファイル体系が存在する。またシステム2への可搬媒体500からのデータの移動は、インデックスデータのみをコピーするものであり（本願発明）、イメージファイルは可搬媒体500上に残っている。したがって、可搬媒体500への移動（コピー）を前提としたインデックス情報によれば、システム1又はシステム2自身のファイル体系に関係なく、常に、可搬媒体500上にイメージデータが存在する形でインデックスを付けることにより上記の問題を解決できる。

【0029】具体的には、インデックスRDBのデータ部分、インデックスDB300の中では、ファイル位置テーブル320の内容は、“IMAGE1”と¥IMAGE¥FOLDER¥img1.xxxxをリンクさせて登録する。入出力領域400内のファイル位置テーブル420（図3）に、このデータを記録する際には¥M01¥img1.xxとして記録する。この記録の際にパスを書き換えることを転送位置変換部490にて実施する。

【0030】この“M01”は可搬媒体上にあることを示す各システム間共通のパス名であり、実際には“M01”以下に多階層の構造を持たせることもできる。この結果、システム2において入出力領域400にコピーされたRDBのデータで、パス名のテーブル320では、“IMAGE1”的パス名として、¥M01¥FOLDER¥img1.xxxxが記録されることになる。

【0031】このため入出力領域400のデータを扱うようにDBを切り替えれば、可搬媒体500上のイメージデータのパスが直接、入出力領域400上のDBからアクセスできることになる。これにより可搬媒体500により、イメージを登録したシステム1から別のシステム2へ、イメージデータとそのインデックス情報を移動する場合にインポート処理を行わずに、可搬媒体500のデータへシステム2がアクセスすることが可能となる。

【0032】次に、システム2においてイメージを登録する際の動作について説明する。イメージを登録する場合、イメージデータをイメージDB100に記録し、そのインデックスデータをインデックスDB300に記録する。しかしながら、該データを可搬媒体500を用いて別のシステムに移動することを前提とする場合は、同

時に入出力領域400にもインデックスを記録する。このとき前述の通り、ユーザ名、セキュリティ情報等の各システムに依存するデータは、両方（システム2及び可搬媒体500）に同一のものを予め設定しておく。

【0033】通常のデータベース管理部であるインデックスDB300は、1つのオペレーションは1つのデータベースに対してのみ実行する。本願発明では、これを複数のDBファイルに同時に実施する機能を持たせることにより本機能を実現している。

【0034】図3を用いてデータ登録の際の各DBとのテーブルへの処理を説明する。図3の410、420、430、450は、それぞれ、入出力領域400におけるテーブルであり、図2の310、320、330、350に対応する。図2の310、320、330、350は、システム2におけるインデックスDB内の各テーブルを意味する。

【0035】両者は構造的には全く同一であり、ユーザ情報に関するテーブル450は350と同じ内容を記録しておく。この入出力領域400のDBに記録されるインデックスは、インデックスDB300（図1）の中の一部であり、蓄積されたインデックスに対応するイメージデータと合わせて可搬媒体500に収まるデータ量であることが必要である。

【0036】イメージ登録の際に、システムにより入出力領域400へも登録すると判断されたときは、タイトルとIDは共にテーブル310と410に同一のデータを書き込む。これに対して、テーブル420は、テーブル320のデータを、転送位置切り替え部490を経由して、これにより変換して記録することで、可搬媒体500におけるパス名を記録することとなる。430のレコードのデータ内容は、410と同様に、330と同一の構造のデータを、該当する部分だけ取り出したデータとして記録する。

【0037】ファイルを可搬媒体500によりシステム2の外に出す処理を、エクスポート処理と呼ぶ。エクスポートの際には、インポートと逆に、インデックスデータは既に媒体上に記録済みであり、イメージデータファイルをイメージDB100から可搬媒体500へコピーすることが必要となる。

【0038】登録の際に入出力領域400にも書き込んだイメージファイルの識別情報は、既知の方法により移動することが可能である。例えば、インデックスDB300に識別テーブル390を設定し、そこに“MO1”と記載されているファイルを検索し、可搬媒体500にコピーする方法が考えられる。

【0039】以上の方法を応用することにより下記のようなシステムを構築することもできる。図4に示すのはデータ登録を専用に行うシステムの1つの構成例である。このシステムでは、登録されたデータを可搬媒体により他のシステムに移動して利用することを前提とする

ものである。そのため図1のシステムと比較して、継続的にインデックスデータを蓄積するインデックスDB300を廃し、入出力領域を400に対応する、複数の入出力領域400、600、700を有している。

【0040】イメージの登録の際には、予め、その内容に合わせて400、600、700のいずれにインデックスを記録するかを定め、インデックスDB管理部200は、登録するDBファイルに接続（アクセス）した状態で登録を行う。

【0041】登録後、当該入出力領域（400、600、700等の複数のもののうちいづれか1つ）と該当するイメージデータファイルを可搬媒体500にコピーする。複数の入出力領域を切り替ながら登録する場合には、可搬媒体も複数用意することになる（可搬媒体500、510）。その場合に上記のインデックスDB300に記録した識別テーブル390の媒体名称は“MO1”だけでなく複数の媒体を識別できるように媒体の名称若しくはこれに代る識別子を記録する。

【0042】一方、複数の可搬媒体により他のシステムで登録されたデータを、読み取る側のシステムも上記と同様に、はじめからデータ登録は複数の可搬媒体により実行することを前提として構築することもできる。

【0043】この場合、複数の入出力領域を持つことで、媒体単位で別々のDBにアクセスすることが最も簡単な構造である。1つのインデックスDB300に可搬媒体の中のインデックス情報をコピーすることで、システムを動作させることも、上記の方法を応用すれば可能である。

【0044】登録用のシステム1でインデックスを作成する場合に、ユニークなデータであることを要求されるデータ、例えば、イメージID等は、利用する媒体毎に先頭文字を換えるなどして、重複を生じないようにする。こうすればデータをシステム2のインデックスDB300にコピーした場合に重複するIDは存在しない。そこで、各DBファイルをテーブル単位に分割し、それを各テーブル毎に合成していくことにより、基本的に1つのDBとすることができる。ただし、このDBは機械的に接続しただけで整合性が取られていない。整合性は、合成後にソート処理することにより解決できる。

【0045】

【発明の効果】本発明によれば、RDBを用いたシステムにおいても、システム間でDBの一部のデータを分割して受け渡し機能させることを、容易に短時間で実行できる。

【図面の簡単な説明】

【図1】本システムの構成の一例を説明するための図である。

【図2】図1のシステムで用いるデータベースの構造を説明するための図である。

【図3】本システムで用いるデータベースにおいて、記載内容の切り替えを説明するための図である。

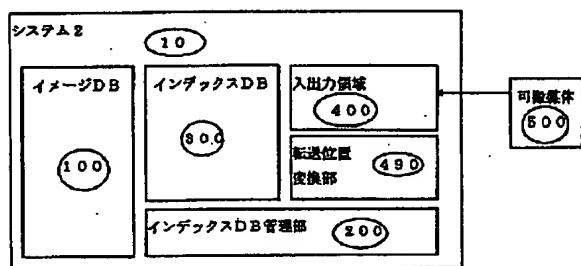
【図4】本発明の一応用例の構成を説明するための図である。

【符号の説明】

100…イメージデータベース、200…インデックスデータベース管理部、
300…データベース本体であるインデックスデータベース、
400…データベースの一部を別のシステムに移動するときに用いる入出力領域、
500…可搬媒体、310…イメージタイトルのテーブル、
320…ファイルの格納位置を示すパス名を記録するテーブル、
330…各ファイル毎にアクセス可能なユーザを記録するセキュリティテーブル、
350…各ユーザを登録するユーザテーブル。

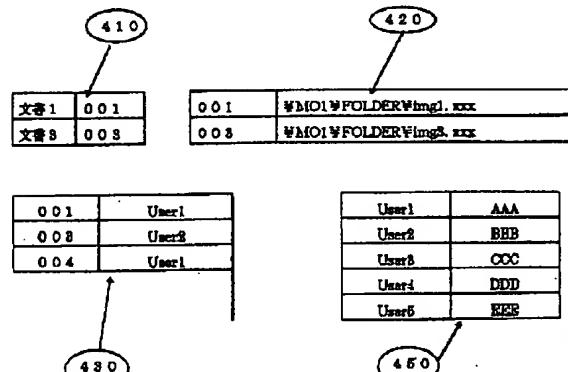
【図1】

図1



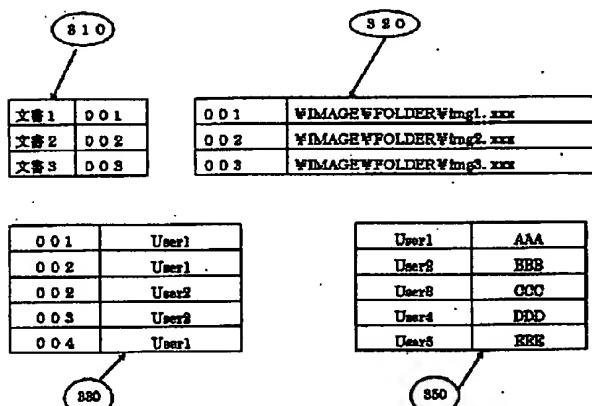
【図3】

図3



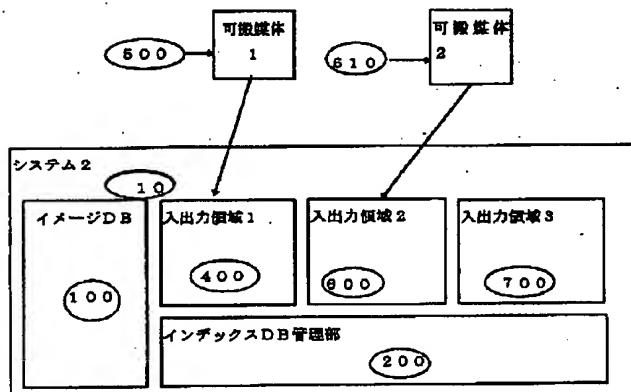
【図2】

図2



【図4】

図4



(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平6-214849

(43)公開日 平成6年(1994)8月5日

(51)Int.Cl.⁵
G 0 6 F 12/00

識別記号 庁内整理番号
5 2 0 A 8526-5B
5 1 2 8526-5B

F I

技術表示箇所

審査請求 未請求 請求項の数 2 O L (全 11 頁)

(21)出願番号 特願平5-4948

(22)出願日 平成5年(1993)1月14日

(71)出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72)発明者 加藤 宣弘

神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

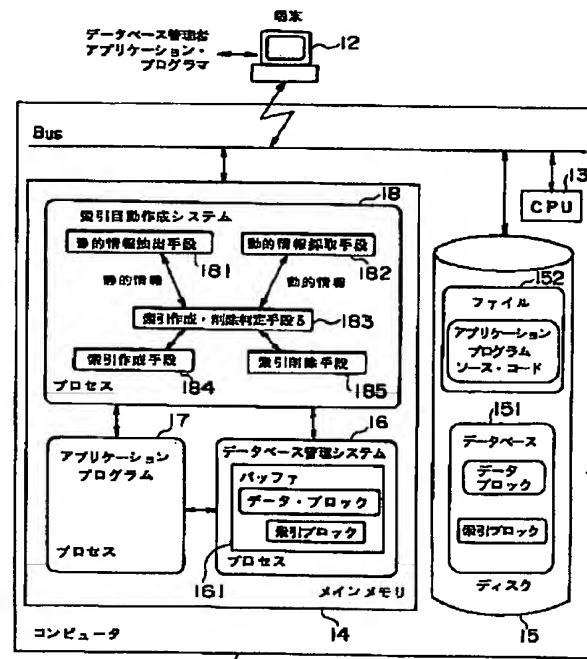
(74)代理人 弁理士 鈴江 武彦

(54)【発明の名称】 データベースシステム

(57)【要約】

【目的】高速なデータ操作を行なうための索引だけを自動作成し、データベース操作の高速化を図る。

【構成】索引自動作成システム18は、アプリケーション・プログラム17のデータ操作手続きの内容を抽出し、その抽出した内容とデータベース151のテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数を求める。そして、そのブロック数に基づいて、索引作成の可否が索引作成・削除判定手段183により判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、無駄な索引がなくなり、常に高速なデータ操作を行なうことが可能になる。



【特許請求の範囲】

【請求項1】 各種アプリケーションプログラムからのデータ操作手続き要求に応じてリレーションナル・データベースのデータ操作を実行するデータベースシステムにおいて、

アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きに関する情報を抽出する手段と、この抽出したデータ操作手続きに関する情報とそのデータ操作手続きで扱われるテーブルの構成情報とに基づいて、データ操作手続きの実行のために2次記憶装置との間で入出力が必要なデータブロック数を、各データ操作手続き毎に算出する手段と、

この算出された各データ操作手続き毎の入出力データブロック数を統合し、その統合結果に従って、索引を作成するか否かを前記リレーションナル・データベースを構成する各テーブル毎に判定する手段と、

この判定結果に従って索引を作成する手段とを具備することを特徴とするデータベースシステム。

【請求項2】 データベースのデータにアクセスするためのデータ操作手続きを含むアプリケーションの実行が可能であり、索引を用いた検索が可能であるデータベースシステムにおいて、

システムを稼動する前に、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きの部分から静的な情報を抽出する静的情報抽出手段と、

システム稼動時にそれぞれのアプリケーションのデータ操作手続きの実行頻度、実行時間などの動的情報を採取する動的情報採取手段と、

上記静的情報抽出手段により抽出された情報を基にそれぞれの表に対する索引を作成すべきどうかを、また上記静的情報抽出手段により抽出された情報を基に表に対する索引を新たに作成するかどうか、あるいは既存の索引を削除するかどうかを判定する索引作成・削除判定手段と、

この判定結果にしたがって索引の作成、および作成された索引の削除を行なう索引作成・削除手段とを具備することを特徴とするデータベースシステム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 この発明はデータベースシステムに関し、特に自動索引作成システムを持つリレーションナルデータベースシステムに関する。

【0002】

【従来の技術】 一般に、データベースのデータは、一定の大きさをそれぞれ持つ複数のデータ・ブロックから構成されている。これらデータ・ブロックは、磁気ディスクなどの外部記憶装置内に格納されている。

【0003】 リレーションナルデータベースにおいて、例えば、あるテーブルから特定のタブルを検索する場合には、検索対象のテーブルの先頭タブルを含むデータ・ブ

ロックを見つけ、ディスクからメモリにそのブロックを読み出す。もしそのブロックに検索対象のタブルが含まれていれば、ディスクからの読み出しは終るが、そのブロックに含まれていなければ、次のブロックを読み出す。このようなデータ・ブロックの読み出し動作は、検索対象のタブルを含むブロックを読み出すまで続けられる。

【0004】 リレーションナルデータベースのテーブルは、多くのタブルから構成されている。データベースのデータを格納するデータ・ブロックの数は、タブルが多いほど増加される。データ・ブロックの数が少なければ、前述のような検索のためのディスクからメモリへのブロックの平均読み出し回数は少なくて済む。しかし、テーブルを構成するタブル数が多くそれらタブルが多くのブロックに格納されている場合には、ブロックの平均読み出し回数は多くなる。そこで、従来より、検索対象のタブルを含むブロックがどれであるかをより少ないブロック読み出し回数で知ることができるような手法を考えられている。

【0005】 その手法の1つが索引である。つまり、テーブルに含まれる各タブルの1つ以上のフィールドの値を用いてバランス木のようなデータ構造を作り、それを索引ブロックとして格納する。索引を用いたタブルの検索では、まずテーブルの先頭の索引ブロックをディスクから読み出し、検索対象のタブルのフィールドの値をキーとして、索引ブロック内を探索する。もし検索対象のタブルが格納されているデータ・ブロックの情報が、その索引ブロック内に存在すれば、その情報を用いてデータ・ブロックを読み出す。その索引ブロックに存在しなければ、索引データ構造に従って探し、次の索引ブロックを読み出し、同様に検索対象のタブルが格納されているデータ・ブロックの情報が存在するかどうか調べる。索引ブロックの読み出しは、検索対象のタブルのデータ・ブロックの情報を見つけるまで行なう。

【0006】 一般に、索引ブロック数はデータ・ブロック数よりも少ない。また、索引ブロックの読み出し順はバランス木のようなデータ構造に基づいているので、一般に索引を用いたタブルの検索は、索引を用いない場合よりも、ブロックの読み出し回数が少なく、高速である。しかし、テーブルを構成するほとんど全てのタブルを検索するような場合には、ほとんど全てのデータ・ブロックを読み出す必要があるので、索引ブロックの読み出しが無駄になり、索引を用いない検索の方が高速になる。また、タブルの挿入、削除や、索引に使われているフィールド値の更新を頻繁に行なう場合には、対応する索引ブロックを頻繁に変更しなければならないので、索引がある場合にこれらの操作に多くの時間を要することになる。

【0007】 従来、システムで実行されるアプリケーションをそれぞれ高速化するために、データベース管理

者、アプリケーション・プログラムなどが、データベースの各テーブルに対して索引を作成していた。しかし、上記のように索引を作成することによりデータ操作が高速になる場合もあれば、そうでない場合もあるので、索引作成に関する知識を必要とした。また従来1つアプリケーション・プログラムを高速にするために、データ操作の実行トレースを採取できるシステムが存在した。しかし、一般にデータベースにアクセスするシステムでは、多数のアプリケーションが起動され、種々雑多なデータ操作が実行されるため、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがあった。

【0008】

【発明が解決しようとする課題】従来のデータベースシステムでは、索引を作成することによりデータ操作が高速になる場合もあれば、逆に遅くなってしまうこともあります、索引を作成するか否かの判断には、データベース管理者等にはアプリケーション・プログラムの内容とデータベースの論理的・物理的構造に関する知識が必要された。

【0009】また、多数のアプリケーションが起動されるシステムでは、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがあります、システム全体としての高い性能を達成するように索引を作成することは困難であった。

【0010】この発明はこのような点に鑑みてなされたものであり、アプリケーション・プログラムおよびデータベースのテーブル内容に基づいて索引作成の可否を自動判定できるようにして、常に高速なデータ操作を行うことができるデータベースシステムを提供することを目的とする。

【0011】

【課題を解決するための手段および作用】この発明は、各種アプリケーションプログラムからのデータ操作手続き要求に応じてリレーショナル・データベースのデータ操作を実行するデータベースシステムにおいて、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きに関する情報を抽出する手段と、この抽出したデータ操作手続きに関する情報とそのデータ操作手続きで扱われるテーブルの構成情報に基づいて、データ操作手続きの実行のために2次記憶装置との間で入出力が必要なデータブロック数を、各データ操作手続き毎に算出する手段と、この算出された各データ操作手続き毎の入出力データブロック数を統合し、その統合結果に従って、索引を作成するか否かを前記リレーショナル・データベースを構成する各テーブル毎に判定する手段と、この判定結果に従って索引を作成する手段とを具備することを第1の特徴とする。

【0012】このデータベースシステムにおいては、アプリケーション・プログラムのデータ操作手続きの内容とテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数が計算により求められ、そのブロック数に基づいて、索引作成の可否が判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、常に高速なデータ操作を行なうことが可能になる。

【0013】また、この発明は、データベースのデータにアクセスするためのデータ操作手続きを含むアプリケーションの実行が可能であり、索引を用いた検索が可能であるデータベースシステムにおいて、システムを稼動する前に、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きの部分から静的な情報を抽出する静的情報抽出手段と、システム稼動時にそれぞれのアプリケーションのデータ操作手続きの実行頻度、実行時間などの動的情報を採取する動的情報採取手段と、上記静的情報抽出手段により抽出された情報を基にそれぞれの表に対する索引を作成すべきどうかを、また上記静的情報抽出手段により抽出された情報を基に表に対する索引を新たに作成するかどうか、あるいは既存の索引を削除するかどうかを判定する索引作成・削除判定手段と、この判定結果にしたがって索引の作成、および作成された索引の削除を行なう索引作成・削除手段とを具備することを第2の特徴とする。

【0014】このデータベースシステムでは、システムを稼動する前にアプリケーションプログラムのデータ操作手続きとデータベースの静的な情報から、テーブルのフィールドに索引を作成するかどうかを判定し、有効な索引だけを作成する。またシステム稼動時の実行頻度、実行時間などの動的な情報から、既存の索引の有効性を検証し、また新たな索引を作成すべきかどうかを判定する。以上により、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性が判定されるので、システムに保持されるのは有効な索引だけになる。したがって、多数のアプリケーションが起動されるシステムでも、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがなくなり、システム全体としての高い性能を達成できる。

【0015】

【実施例】以下、図面を参照してこの発明の実施例を説明する。図1には、この発明の一実施例に係わるデータベースシステムを実現するためのコンピュータシステムの構成が示されている。

【0016】このデータベースシステムは、ホストコンピュータ11およびこれに接続された端末12を含むコンピュータシステムによって実現される。ホストコンピュータ11は、CPU13、メインメモリ14、および磁気ディスク装置15を備えている。CPU13は、こ

のホストコンピュータ11全体の動作を制御するためのものであり、メインメモリ14に格納されたプログラムの実行によって、データベース処理に係わる種々の演算、および磁気ディスク装置15の入出力制御を初め、索引自動作成のための処理等実行する。

【0017】メインメモリ14には、データベース管理システム16、アプリケーションプログラム17、および索引自動作成システム18がそれぞれ実行対象のプロセスとして格納される。

【0018】データベース管理システム16は、SQLのようなデータ操作言語を用いて磁気ディスク装置15のリレーションナルデータベース151の検索、更新等の各種のリレーションナルデータベース演算を実行するものであり、アプリケーションプログラム17からの要求に従って動作される。また、データベース管理システム16は、ビュー機能をサポートする。このビュー機能は、端末12を操作するデータベース管理者に対して様々な視点のテーブルの視像を画面表示する機能である。バッファ161に読み出されたリレーションナルデータベース151のテーブルはその論理構造が異なるビューに変換され、それが端末12に画面表示される。データベース管理者やアプリケーションプログラムは、画面表示されたビューを参照しながらデータベースの検索等を行なうことができる。

【0019】アプリケーションプログラム17は、データベース管理システム16を利用した各種データベース操作を実行する。このアプリケーションプログラム17は、実行時に磁気ディスク装置15のファイル152からメインメモリ14にロードされる。

【0020】索引自動作成システム18は、アプリケーション・プログラム17およびデータベース151のテーブル内容に基づいて索引作成の可否を自動判定し、索引により高速なデータ操作を行なうことができる時には、索引を自動作成する。この索引自動作成システム18は、静的情報抽出手段181、動的情報採取手段182、索引作成・削除判定手段183、索引作成手段184、および索引削除手段185から構成されている。

【0021】静的情報抽出手段181は、システムを稼動する前、具体的には磁気ディスク装置15のファイル152からメインメモリ14にアプリケーション・プログラム17をロードする時に、そのアプリケーションプログラムのデータ操作手続きの部分からデータ操作の種類、アクセスするテーブル、集約関数と集約関数を適用するフィールド、更新するフィールド、検索フィールドと検索条件を抽出する。

【0022】動的情報採取手段182は、システム稼動時に、その時に実行されているアプリケーション・プログラム17のデータ操作手続きの実行頻度、実行時間を採取する。

【0023】索引作成・削除判定手段183は、静的情

報抽出手段181により抽出された情報を基にそれぞれのテーブルに対する索引の候補が作成条件を満たすかどうかを判定する。また索引作成・削除判定手段183は、動的情報採取手段182により採取された情報を基に、それぞれのテーブルに対する索引を見直し、索引の作成、削除を判定する。索引作成手段184、および索引削除手段185は、索引作成・削除判定手段183の指示を受けて、それぞれ索引を作成、削除する。図2にテーブルの一例を示す。

【0024】ここでは、データベース151に、従業員テーブル(EMP)T1、所属テーブル(DEPT)T2、物品発注・納入テーブル(ORDER)T3が定義されている場合を考える。

【0025】従業員テーブル(EMP)T1は従業員に関する情報を保持するテーブルであり、所属テーブル(DEPT)T2は従業員の所属部署に関する情報を保持するテーブル、物品発注・納入テーブル(ORDER)T3は従業員が発注した物品の納入状況に関する情報を保持するテーブルである。

【0026】いま、従業員テーブル(EMP)T1には1000レコード、所属テーブル(DEPT)T2には10レコード、物品発注・納入テーブル(ORDER)T3には10000レコードがそれぞれ含まれているとする。各フィールド名の右横に示した数は、フィールドのバイト数を表す。

【0027】例えば、従業員テーブル(EMP)T1においては、従業員番号(EMP_NO)フィールドは4バイト、従業員名(EMP_NAME)フィールドは20バイト、従業員に関するその他の情報(EMP_ELS)のためのフィールドは100バイト、従業員番号(DEPT_NO)は4バイトから構成されている。

【0028】従って、従業員テーブル(EMP)T1のレコードの大きさは128バイトとなる。同様に、所属テーブル(DEPT)T2のレコードの大きさは124バイト、物品発注・納入テーブル(ORDER)T3のレコードの大きさは132バイトとなる。図3にデータ・ロックの例を示す。

【0029】1つのデータ・ロックのサイズは2048バイトである。ロックの先頭の100バイトはロック・ヘッダとして確保され、最後の200バイトは拡張領域として確保されている。残りの1748バイトがデータを格納するために用いられる。このデータ格納領域においては、各レコード毎に10バイトのレコード・ヘッダが確保されている。従って、1レコード・サイズがLバイトの場合には、1データ・ロックに、 $1748 / (L + 10)$ の数のレコードが格納できることになる。ゆえに、従業員テーブル(EMP)T1、所属テーブル(DEPT)T2、物品発注・納入テーブル(ORDER)T3のデータ・ロックにはそれぞれ最大12、13、12レコードが格納できる。

【0030】いま従業員テーブル (EMP) T 1、所属テーブル (DEPT) T 2、物品発注・納入テーブル (ORDER) T 3にはそれぞれ 1000、10、1000 レコードが含まれているので、それぞれ 84、1、834 ブロックの領域を必要とする。図4に各テーブルに対するレコードあたりのバイト数、ブロックあたりのレコード数、ブロック数を示す。図5に索引ブロックの例を示す。

【0031】索引ブロックにおいては、データ・ブロックと同様に、1748 バイトが索引レコードを格納するために用いられる。索引を付けるフィールドの値とそれに対応するレコードの格納位置を示すポインタ (20 バイト) が、1つの索引レコードを形成する。例えば従業員テーブル (EMP) T 1 のフィールド (EMP_NO) に索引を付けるとすると、1つの索引データに $4 + 20 = 24$ バイト必要となり、1 索引ブロックに 72 索引レコードを格納できる。テーブル T 1 には 1000 レコードがあるので、すべてのレコードに対する索引を作るためには、14 個の索引ブロックが必要となる。また、索引をバランス木で構成する場合には、バランス木の構成を格納するための索引構造ブロックが必要となる。索引構造ブロックには、索引ブロック数の約 1 割が必要であるので、上記の場合には 1 個の索引構造ブロックが必要となり、合わせて 15 ブロックを索引のために確保する必要がある。

【0032】図6に、各テーブルの主なフィールドに対する索引ブロック数、索引構造ブロック数を示す。ここで、所属テーブル (DEPT) T 2 のフィールド DEPT_NO に対して索引構造ブロックがないのは、索引構造が索引ブロックに含まれているからである。

【0033】図7には、図2のテーブル T 1 ~ T 3 に対して実行されるアプリケーション・プログラム 17 の一例が示されている。図7 (a) はある従業員が物品を発注するときのプログラム、(b) は納入担当者が物品を納入するときのプログラム、(c) はある部所全体の注文、納入の状況を表示するためのプログラムである。プログラムにおいて英小文字はホスト言語とデータ操作言語とで共用される変数である。または[]は配列を表す。以下、それぞれのプログラムについて簡単に説明する。

【0034】発注プログラムは、物品発注・納入テーブル (ORDER) T 3 の更新等を行なうためのものであり、この発注プログラムでは、まず入力された従業員番号と発注品名をそれぞemp_noとorder_nameにセットする。そして、物品発注・納入テーブル (ORDER) T 3 からフィールド (ORDER_NO) の最大値を検索し、それを 1だけインクリメントしてorder_noにセットする。そして、テーブル (EMP) T 1 からフィールド EMP_NO の値がemp_noに等しいレコードを検索し、フィールド (DEPT_NO) の値をdept_noにセ

ットする。最後に新しいレコードをテーブル (ORDER) T 3 に入力する。

【0035】納入プログラムは、テーブル (ORDER) T 3 のフィールド (FLAG) を更新するためのものであり、この納入プログラムでは、入力された発注品番号 (納入品番号) をorder_noにセットし、テーブル (ORDER) T 3 からフィールド (ORDER_NO) の値がdept_noに等しいレコードを検索し、そのフィールド (FLAG) の値を 1 に更新する。

【0036】状況表示プログラムは、発注品名や納入状況等を画面表示するためのものであり、この状況表示プログラムでは、まず入力された部所番号をdept_noにセットし、テーブル (DEPT) T 2 からフィールド DEPT_NO の値がdept_noに等しいレコードを検索し、フィールド DEPT_NAME の値をdept_nameにセットする。次に、テーブル (ORDER) T 3 からフィールド DEPT_NO の値がdept_noに等しいレコードを検索し、フィールド ORDER_NAME、FLAG の値を配列order_name[], flag[] にそれぞれセットする。ここで配列を用いるのは、この問合せでは複数のレコードを返される可能性があるからである。最後に、dept_name、order_name[], flag[] を表示する。以上で説明したテーブルとアプリケーション・プログラムに対して本発明を適用した場合について以下説明する。

【0037】静的情報抽出手段 181 は、図7に示すアプリケーション・プログラム 17 から図8に示す情報を抽出する。図8には、それぞれのデータ操作手続きに対して、その識別子、手続きが含まれるアプリケーション・プログラム、その命令の種類、アクセスするテーブル名、値を更新するフィールド名、利用する集約関数、集約関数を適用するフィールド名、検索に使われるフィールド名、検索条件が含まれる。図8において、横線が引かれている箇所は対応する項目がないことを表す。更新フィールド名の * はすべてのフィールドを更新することを表す。検索条件の = val は、ある値に等しい値を持つフィールドを含むレコードだけを選択することを表す。

【0038】システムを稼動する前に索引作成・削除判定手段 183 は、図8に示す情報から各データ操作手続きに対するレコードの選択率を次のように計算する。ここで選択率とは、データ操作手続きにより選択したレコード数の全レコード数に占める割合である。

【0039】図9に各データ操作手続きに対する選択率を示す。ただし、レコードを選択しないデータ操作手続きに対しては横線を引いておく。レコードの選択率は次のように求める。例えば、テーブル (EMP) T 1 のフィールド DEPT_NO に対するデータ操作手続き 2 の選択率は次のようにして求めることができる。1000人の従業員の中から特定の従業員 1人を選択するから、選択率は $1 / 1000 = 0.001$ となる。

【0040】さらに、索引作成・削除判定手段 183

は、図4、図6、図8、図9に示す情報から、索引を作成しない場合のブロック読み出し数と索引を作成する場合のブロック読み出し回数を計算する。例えば、データ操作手続き1に対して、索引を作成しない場合、および索引を作成する場合それぞれのブロック読み出し回数を求める。索引を作成しない場合にはテーブル(ORDER)T3のすべてのデータ・ブロックを読まなければならないので、834ブロックとなる。一方、索引を作成する場合には、索引構成ブロックを14ブロック、索引ブロックを1ブロック、データ・ブロックを1ブロック、合計16ブロックだけ読めばよい。したがって、テーブル(ORDER)T3のフィールドDEPT_NOに対する索引を作成すると仮定する。

【0041】また、データ操作手続き6に対して同様に計算してみると、索引を作成しない場合には、すべてのデータ・ブロックを読まなければならないので、834ブロックとなる。一方、索引を作成する場合には次のように計算できる。

【0042】テーブル(ORDER)T3は10000レコードを含んでおり、データ操作手続き6の選択率は0.1なので、1000レコードを選択することになる。索引ブロックにおいて、索引レコードはDEPT_NOの値の順にソートされているので、あるDEPT_NOの値の索引レコードは14ブロックに保持されている。データ・ブロックにおいて、レコードはランダムに配置されているので、あるDEPT_NOの値に対する100レコードは583ブロックに配置されている。ゆえに、索引を作成する場合には、索引構造ブロックを14ブロック、索引ブロックを14ブロック、データ・ブロックを583ブロック、合計611ブロック読まなければならない。したがって、テーブル(ORDER)T3のフィールドDEPT_NOに対する索引を作成すると仮定する。

【0043】図10にそれぞれのデータ操作手続きに対して、索引を作成しない場合、作成する場合の読み出しブロック数、索引を作成するか否かを示す。データ操作手続き3は、テーブル(ORDER)T3のすべてのフィールドを更新するので、それらのフィールドに対して索引を作成しないと仮定する。データ操作手続き3(INSET命令)に対しては、入出力(読み出しと書き込み)ブロック数を示す。索引がない場合には空き領域のあるブロックを読み書きするので、入出力ブロック数は2となる。索引がある場合には、それに加えて索引ブロック、索引構造ブロックの修正のために入出力ブロック数が多くなる。テーブル(ORDER)T3にはフィールドORDER_NOとDEPT_NOに索引を作成する可能性があるので、最大索引構造ブロックを14ブロック、索引ブロックを2ブロックを読み出し、書き込むことになる。したがって、合計 $2 + (14+2)*2 = 34$ となる。

【0044】索引作成・削除判定手段183は、図8、図10からテーブルのフィールドに索引を作成するかどうかを判定する。テーブル(ORDER)T3のフィールドORDER_NO、DEPT_NOに対しては、索引を作成しないという仮判定と索引を作成するという仮判定がある。各データ操作手続きが同じ頻度で実行されるならば、図10よりフィールドORDER_NO、DEPT_NOのそれぞれにおいて、索引を作成する場合は索引を作成しない場合よりも入出力ブロック数が少ない。ゆえに、それぞれ索引を作成すると判定する。テーブルEMPのフィールドEMP_NOには索引を作成すると判定し、テーブルDEPTのフィールドDEPT_NOには索引を作成しないと判定する。索引作成手段184は、これらの判定に基づいて索引を作成する。なお、静的情報による索引の作成判定において、入出力ブロック数だけでなく、各種オーバヘッドの見積りを含めてもよい。

【0045】動的情報採取手段182は、システム稼動時に、実行されるアプリケーションプログラムによる実際のデータ操作手続きの実行頻度、平均実行時間を採取し、記憶する。いま図11に示すような実行頻度と平均処理時間が得られたとする。

【0046】索引作成・削除判定手段183は、各データ操作手続きに対して、図11に示す実行頻度を図10の入出力ブロック数に乗じる。その結果を図12に示す。図12に示す値は各データ操作手続きが1秒あたりに入出力するブロック数を表す。この値を用いて、上記と同じように索引を作成するか否かの判定を行なう。

【0047】テーブル(ORDER)T3のフィールドORDER_NOに索引を作成するか否かで影響を受けるのは、データ操作手続き1、3、4、である。索引を作成しない場合には、それらの入出力ブロック数の和が11864になるが、作成する場合には577となる。したがって、テーブル(ORDER)T3のフィールドORDER_NOには索引を作成すると判定する。テーブル(ORDER)T3のフィールドDEPT_NOに索引を作成するか否かで影響を受けるのは、データ操作手続き3と6である。索引を作成しない場合には、それらの入出力ブロック数の和が1022になるが、作成する場合には1083となる。したがって、テーブル(ORDER)T3のフィールドDEPT_NOには索引を作成しないと判定する。この判定は、静的情報から求めた上記の判定とは異なるので、索引削除手段185に対して既に作成されている索引の削除を指示する。

【0048】索引を見直した後、動的情報採取手段182は、データ操作手続きの実行頻度、平均実行時間を採取し、記憶する。索引作成・削除判定手段183は、実行頻度と平均実行時間との積のデータ操作手続きに対する総和を、索引の見直しの前後で比較する。もし見直し後の値の方が大きければ、索引を見直し以前の状態に戻

す。

【0049】なお、各データ操作手続きに対する入出力ブロック数は、静的情報から得られた値ではなく、動的情報から得られた値を採用してもよい。その場合には、動的情報採取手段182は、システム稼動時にデータ操作手続きの平均入出力ブロック数も採取する。また、索引の見直し入出力ブロック数ではなく、平均実行時間を使ってもよい。そのためには、索引がある場合とない場合についてデータ操作手続きの実行時間を採取する必要がある。次に、図13のフローチャートを参照して、索引作成の手順を説明する。

【0050】まず、静的情報抽出手段181は、実行対象のアプリケーションプログラムのデータ操作手続きの部分から静的情報を抽出する(ステップS11)。次いで、索引作成・削除判定手段183は、静的情報とデータ操作手続きの選択率に基づき、各データ操作手続きの入出力ブロック数を計算する(ステップS12)。そして、索引作成・削除判定手段183は、各データ操作手続きの入出力ブロック数に基づき、索引を作成するか否かを仮判定する(ステップS13)。

【0051】この後、索引作成・削除判定手段183は、索引作成の仮判定結果を統合し索引を作成するかどうかを判定する(ステップS14)。そして、索引作成手段184は、判定結果に従って、必要なテーブルの索引を作成する。以上、ステップS11～S15の処理はアプリケーションプログラムの実行前に行なわれる処理である。この後、アプリケーションプログラムの実行時には、次のステップS16～S18の処理が繰り返し実行される。

【0052】動的情報採取手段182は、各データ操作手続きの実行頻度、実行時間を測定する(ステップS16)。次いで、索引作成・削除判定手段183は、各データ操作手続きの実行頻度、実行時間からなる動的情報に基づき、索引の見直しを行なう(ステップS17)。そして、索引作成手段184および索引削除手段185は、それぞれ見直し結果に基づいて、索引の作成・削除を行なう(ステップS18)。

【0053】以上のように、この実施例においては、アプリケーション・プログラム17のデータ操作手続きの内容とテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数が計算により求められ、そのブロック数に基づいて、索引作成の可否が判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、常に高速なデータ操作を行なうことが可能になる。

【0054】また、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性が判定されるので、システムに保持されるのは有効な索引だけになる。したがって、多数のアプリケーションが起動されるシステムでも、あるアプリケーションの実行を高速化するために作

成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがなくなり、システム全体としての高い性能を達成できる。尚、この発明は、ここに示した実施例に限定されるものではなく、その請求範囲を逸脱しない範囲で適宜変更可能である。

【0055】

【発明の効果】以上のようにこの発明によれば、データベース管理者の手を煩わせることなく、テーブルに有効な索引を作成できる。しかも、アプリケーション・プログラム全体を見通して索引作成の可否を判定するので、システム全体の性能向上を見込める。また、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性を判定するので、システムに保持されるのは有効な索引だけに淘汰されていく。

【図面の簡単な説明】

【図1】この発明の一実施例に係るデータベースシステムの構成を示すブロック図。

【図2】同実施例のデータベースシステムで扱われるデータベースのテーブルの一例を示す図。

【図3】同実施例のデータベースシステムによって入出力されるデータ・ブロックの一例を示す図。

【図4】図3に示した各テーブルに対するレコードあたりのバイト数、レコード数、ブロックあたりのレコード数、ブロック数を示す図。

【図5】同実施例のデータベースシステムによって入出力される索引ブロックの一例を示す図。

【図6】図3に示した各テーブルの主なフィールドに対する索引ブロック数、索引構造ブロック数を示す図。

【図7】同実施例のデータベースシステムで実行されるアプリケーション・プログラムの一例を示す図。

【図8】図7のアプリケーション・プログラムのデータ操作手続きから得られる静的情報の一例を示す図。

【図9】図7のアプリケーション・プログラムの各データ操作手続きによるレコード選択率を示す図。

【図10】図7のアプリケーション・プログラムの各データ操作手続きに対する入出力ブロック数と仮判定結果を示す図。

【図11】同実施例のデータベースシステムで採取される動的情報の例を示す図。

【図12】同実施例のデータベースシステムで採取される動的情報を考慮した場合の各データ操作手続きに対する入出力ブロック数を示す図。

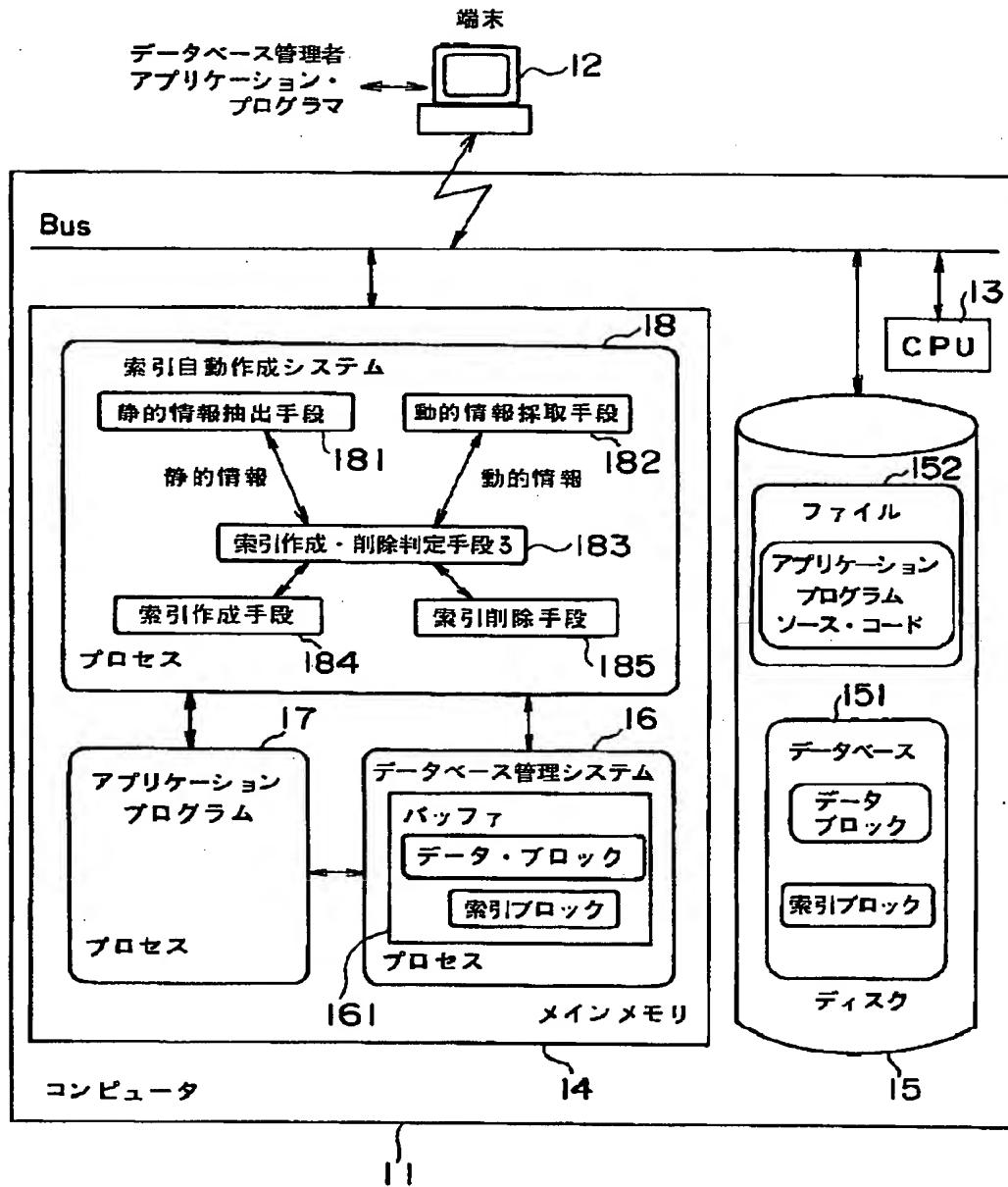
【図13】同実施例のデータベースシステムにおける索引自動作成処理を説明するフローチャート。

【符号の説明】

11…ホストコンピュータ、12…端末、13…CPU、14…メインメモリ、15…磁気ディスク装置、16…データベース管理システム、17…アプリケーションプログラム、18…索引自動作成システム、181…静的情報抽出手段、182…動的情報採取手段、183

…索引作成・削除判定手段、184…索引作成手段、185…索引削除手段。

【図1】



【図4】

テーブル名	レコードあたりの バイト数	ブロックあたりの レコード数	レコード数	ブロック数
EMP	138	12	1000	84
DEPT	134	13	10	1
ORDER	142	12	10000	834

【図6】

テーブル	フィールド	索引レコード あたりの バイト数	ブロック あたりの レコード数	レコード数	索引構造 ブロック数	索引 ブロック数
EMP	EMP_NO	4+20	72	1000	14	1
EMP	DEPT_NO	4+20	72	1000	14	1
DEPT	DEPT_NO	4+20	72	10	1	0
ORDER	ORDER_NO	4+20	72	10000	139	14
ORDER	DEPT_NO	4+20	72	10000	139	14

【図2】

従属性テーブル
(EMP) 4B 20B 100B 4B

EMP_NO(4)	EMP_NAME(20)	EMP_ELSE(100)	DEPT_NO(4)
1	Mike	ABCDE	7
1000	Jane	FGHIJ	3

所属テーブル
(DEPT) 4B 20B 100B

DEPT_NO(4)	DEPT_NAME(20)	DEPT_ELSE(100)
1	Laboratory	KLMNO
10	Sales	PQRST

物品追注・納入テーブル
(ORDER) 4B 20B 4B 100B 4B

ORDER_NO(4)	ORDER_NAME(20)	FLAG(4)	ORDER_ELSE(100)	DEPT_NO(4)
1	Desk	0	UVWXY	4
10000	Chair	1	ZABCD	9

【図3】

データ操作手順表
識別子

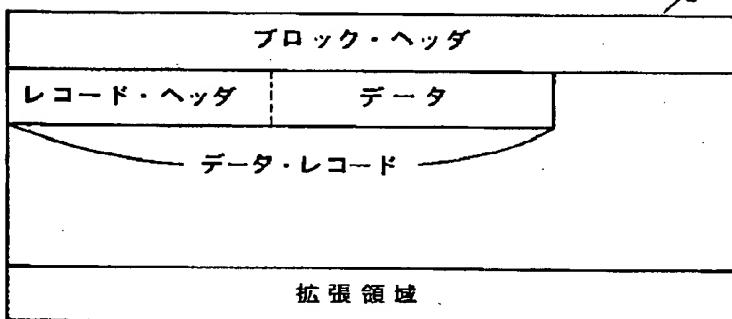
データ操作手順表 識別子	運営率
1	0.0001
2	0.001
3	
4	0.0001
5	0.1
6	0.1

【図4】

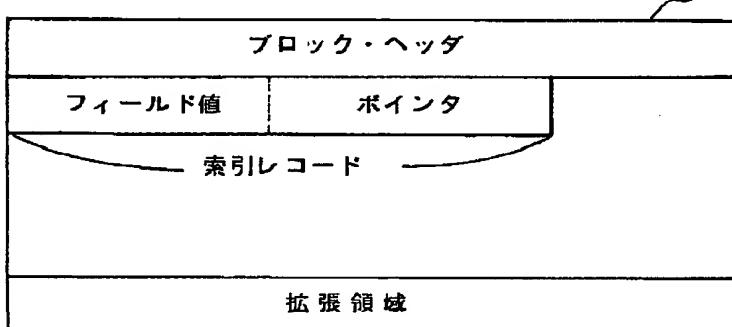
データ操作手順表
識別子

データ操作手順表 識別子	実行頻度 (/sec)	実行時間 (sec)
1	10.3	0.21
2	10.3	0.15
3	10.3	1.20
4	3.9	0.27
5	1.2	0.13
6	1.2	3.78

【図5】



索引ブロック



【図 7】

興 住	INPUT emp_no,order_name SELECT MAX(ORDER_NO)+1 INTO order_no FROM ORDER; SELECT DEPT_NO INTO dept_no FROM EMP WHERE EMP_NO = emp_no; INSERT INTO ORDER VALUES(order_no,order_name, 0,dept_no);
-----	--

(a)

表示	INPUT dept_no SELECT DEPT_NAME INTO dept_name FROM DEPT WHERE DEPT_NO = dept_no; SELECT ORDER_NAME, FLAG INTO order_name[], flag[] FROM ORDER WHERE DEPT_NO = dept_no; DISPLAY dept_name[],order_name[],flag[]
----	---

(c)

新 入

INPUT order_no UPDATE ORDER SET FLAG = 1 WHERE ORDER_NO = order_no;
--

(b)

【図 12】

データ操作 手続名 識別子	入出力ブロック数 (/sec)	
	索引がない場合	索引がある場合
1	8590	165
2	865	31
3	21	350
4	3253	62
5	1	2
6	1001	733

【図 8】

データ操作 手続名 識別子	プログラム	コマンド	テーブル	集約関数	集約関数 適用 フィールド	更新 フィールド	探索 フィールド	探索条件
1	免 注	SELECT	ORDER	MAX	ORDER_NO	—	—	—
2	免 注	SELECT	EMP	—	—	—	EMP_NO	=val
3	免 注	INSERT	ORDER	—	—	*	—	—
4	新 入	UPDATA	ORDER	—	—	FLAG	ORDER_NO	=val
5	表 示	SELECT	DEPT	—	—	—	DEPT_NO	=val
6	表 示	SELECT	ORDER	—	—	—	DEPT_NO	=val

【図 10】

データ操作 手続名 識別子	入出力ブロック数		仮 判定
	索引がない場合	索引がある場合	
1	834	16	作成する
2	84	3	作成する
3	2	34	作成しない
4	834	16	作成する
5	1	2	作成しない
6	834	611	作成する

【図13】

